Component Techniques

Overview

In this lab you'll enhance the "library" web page from the previous lab so that it defines lifecycle methods (for a class-based component) or effect hooks (for a functional component).

Source folders

- ReactDev\Student\06-ComponentTechniques
- ReactDev\Solutions\06-ComponentTechniques

Roadmap

Here's a brief summary of the exercises in this lab. More detailed instructions follow later in this lab document:

1. Familiarization with the 'solution' web pages
2. Getting started with the 'student' web page
3. Storing and retrieving the 'likes' count in local storage

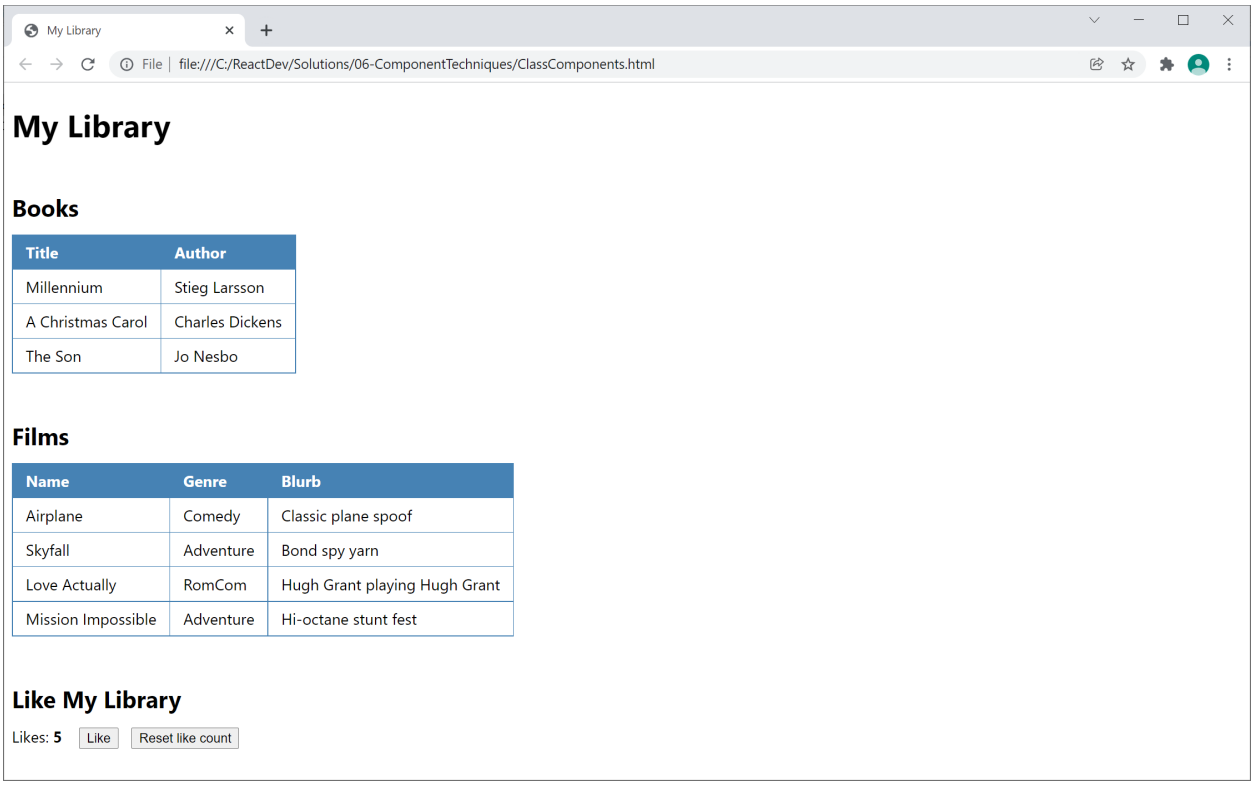
Exercise 1: Familiarization with the 'solution' web pages

Open either of the following web pages in a browser:

ReactDev\Solutions\06-ComponentTechniques\ClassComponents.html

ReactDev\Solutions\06-ComponentTechniques\FunctionalComponents.html

These two web pages are semantically equivalent, and the UI is the same as in the previous lab:



Exercise 2: Getting started with the 'student' web page

Now go to the ***student*** folder and open either of the following web pages in a text editor:

ReactDev\Student\06-ComponentTechniques\ClassComponents.html

ReactDev\Student\06-ComponentTechniques\FunctionalComponents.html

These files are the same as the solutions from the previous lab. Take a moment to get familiar.

Exercise 3: Storing and retrieving the 'likes' count in local storage

Enhance the LikePanel component so that it stores and retrieves the 'likes' count in local storage. Local storage is an HTML5 feature that enables you to store state persistently on the local file system (similar to cookies).

Imagine you have a variable named likes. You can save it in local storage as follows:

```
window.localStorage.likes = likes
```

You can read the likes value from local storage as follows:

```
let likes = window.localStorage.likes
```

So, let's do it… If you're working in ClassComponents.html, follow these steps:

- Implement componentDidMount(), which is called when the component is loaded. Try to get the 'likes' count value from local storage. If the value exists, copy the value into the component's state.
- Implement shouldComponentUpdate(), which is called when React needs to know whether it should re-render the component. Return true if the current 'likes' count is different than the previous value, otherwise return false.
- Implement componentDidUpdate(), which is called after React has re-rendered the component. Write the current value of the 'likes' count to local storage. In this way, local storage stays in sync with the current value of the 'likes' count in memory.

If you're working in FunctionalComponents.html, follow these steps instead:

- Define an effect hook that is only invoked the first time the component is loaded. Try to get the 'likes' count value from local storage. If the value exists, copy the value into the component's state.
- Define an effect hook that is conditionally invoked if the value of the 'likes' count has changed. Write the current value of the 'likes' count to local storage. In this way, local storage stays in sync with the current value of the 'likes' count in memory.

After you've made these changes, your LikePanel component should remember the 'likes' count when you close the web page and then reopen it later. Verify this is what happens.